Лабораторная работа № 8 по курсу дискретного анализа: Жадные алгоритмы

Выполнил студент группы М8О-308б-22 МАИ *Караев Тариел Жоомартбекович*.

Условие

Бычкам дают пищевые добавки, чтобы ускорить их рост. Каждая добавка содержит некоторые из N действующих веществ. Соотношения количеств веществ в добавках могут отличаться. Воздействие добавки определяется как $c_1 \times a_1 + c_2 \times a_2 + \ldots + c_n \times a_n$, где a_i — количество i-го вещества в добавке, c_i — неизвестный коэффициент, связанный с веществом и не зависящий от добавки. Чтобы найти неизвестные коэффициенты c_i, Биолог может измерить воздействие любой добавки, использовав один её мешок. Известна цена мешка каждой из M ($M \geq N$) различных добавок. Нужно помочь Биологу подобрать самый дешевый набор добавок, позволяющий найти коэффициенты c_i. Возможно, соотношения веществ в добавках таковы, что определить коэффициенты нельзя.

Метод решения

Жадные алгоритмы применимы в том случае, если принятие наиболее оптимального решения на каждом шаге решения задачи означает наиболее оптимальное решение задачи в целом. К этой задаче можно применить жадный алгоритм, поскольку чтобы её решить, мы должны отобрать ровно N добавок, а значит эти добавки должны быть наиболее дешёвыми. Идея решения в том, чтобы привести матрицу, составленную из соотношений веществ (для этого вызывается функция SubtractRows), к ступенчатому виду, при этом наверх продвигать строки, характеризующие наиболее дешёвые добавки (строчка ищется функцией FindLowestPriceRow), тогда N верхних строк и будут ответом к задаче. Для получения ответа, необходимо сохранять номера добавок. Таким образом, алгоритм заключается в прохождении по всем добавкам, вызывая функции FindLowestPriceRow и выдвижении найденой строчки вперед и вычитанию этой строчки из последующих, для приведения матрицы к ступенчатому виду

Итоговая сложность $O(M \times N + N^2 \times M) = O(N^2 \times M)$.

Описание программы

Код программы:

```
#include <algorithm>
#include <iostream>
#include <vector>

constexpr const int MAX_NUM = 50;
```

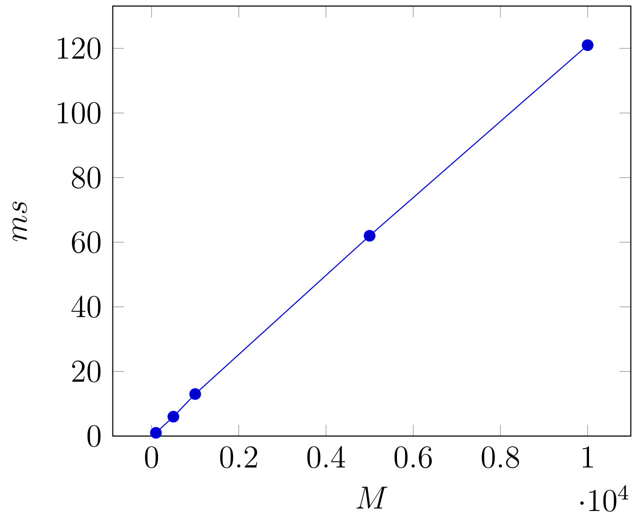
```
struct Addition {
    Addition(size_t n) : ratios(n) {}
    std::vector<double> ratios;
    int price;
    int index;
};

int FindLowestPriceRow(std::vector<Addition> &v, int t) {
    size_t m = v.size();
    size_t n = v[0].ratios.size();
    int minPrice = MAX_NUM + 1;
    int index = -1;
    for (int i = t; i < m; ++i) {
        if ((v[i].ratios[t] != 0.0) && (v[i].price < minPrice)) {
            index = i;
            minPrice = v[i].price;
        }
    }
    return index;
}

void SubtractRows(std::vector<Addition> &v, int t) {
    size_t m = v.size();
    size_t n = v[0].ratios.size();
    for (int i = t + 1; i < m; ++i) {
        double coeff = v[i].ratios[t] / v[t].ratios[t];
        for (int j = t; j < n; ++j) {
            v[i].ratios[j] -= v[t].ratios[j] * coeff;
        }
    }
}

std::vector<int> Solve(std::vector<Addition> &additions) {
    size_t m = additions.size();
    size_t n = additions[0].ratios.size();
    std::vector<int> res;

    for (int i = 0; i < n; ++i) {
        int index = FindLowestPriceRow(additions, i);
        if (index == -1) {
            return {};
```

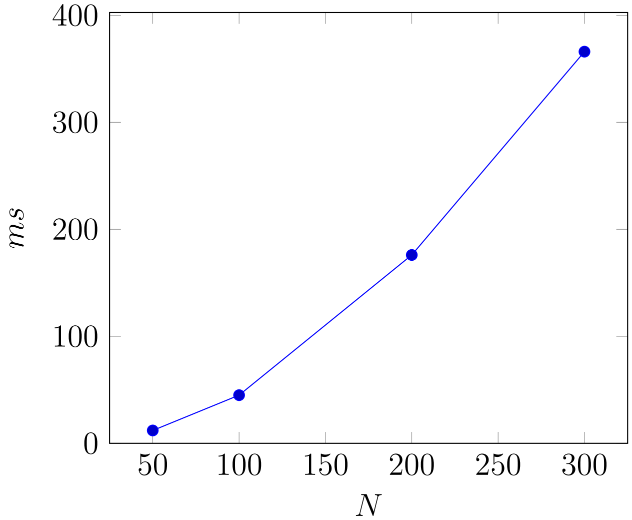
```
        }

        std::swap(additions[i], additions[index]);
        res.push_back(additions[i].index);
        SubtractRows(additions, i);
    }

    std::sort(res.begin(), res.end());
    return res;
}


int main() {
    int n, m;
    std::cin >> m >> n;
    std::vector<Addition> additions(m, Addition(n));

    for (int i = 0; i < m; ++i) {
        for (int j = 0; j < n; ++j) {
            std::cin >> additions[i].ratios[j];
        }
        std::cin >> additions[i].price;
        additions[i].index = i;
    }

    std::vector<int> res = Solve(additions);

    if (res.empty()) {
        std::cout << "-1\n";
        return 0;
    }

    for (auto r : res) {
        std::cout << r + 1 << ' ';
    }
    std::cout << '\n';
    return 0;
}
```

Дневник отладки

Ошибок не обнаружено.

Тест производительности





Приведен тест времени работы алгоритма. По оси X — количество добавок при фиксированном количестве веществ, по оси Y — время выполнения алгоритма в мс (меньше — лучше). На втором графике изменяется количество веществ при фиксированном количестве добавок.

Сложность алгоритма $O(N^2 \times M)$ показана графически

Выводы

Жадные алгоритмы – это алгоритмы, которые на каждом этапе выбирается локально оптимальное решение, рассчитывая на то, что и решение всей этой задачи окажется оптимальным. Многие задачи могут быть успешно решены с помощью жадных алгоритмов, причем быстрее, чем другими методами.

В этой задаче я вспомнил линейную алгебру, а точнее метода Гаусса, а так же разобрался с тем, что составление оптимального набора линейно независимых строк – это задача, которая может быть решена жадным алгоритмом.